

Dialogue management: Function approximation for dialogue policy optimisation

Milica Gašić

Dialogue Systems and Machine Learning Group,
Heinrich Heine University Düsseldorf

Problems in applying RL to dialogue

Gaussian process model for Q -function

GP-Sarsa algorithm

Deep reinforcement learning

Applying reinforcement learning to dialogue

Problems in solving dialogue as an RL task

1. Size of the optimisation problem
 - ▶ Belief state is large and continuous
 - ▶ Set of system actions also large
2. Knowledge of the environment, in this case the user
 - ▶ We do not have transition probabilities
 - ▶ Where do rewards come from?
3. RL algorithms take a long time to converge

Solutions

- ▶ Learn in reduced summary space (1)
- ▶ Learn in interaction with a simulated user (2&3)

Are these good solutions?

Theory: Reinforcement learning

Policy deterministic $\pi: \mathcal{B} \rightarrow \mathcal{A}$
or stochastic $\pi: \mathcal{B} \times \mathcal{A} \rightarrow [0, 1]$

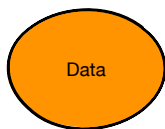
Return $R_t = \sum_{k=t+1}^T \gamma^{k-t-1} r_k$

Q-function What is the value of taking action a in belief state \mathbf{b} under a policy π ?

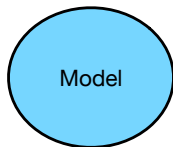
$$Q_{\pi}(\mathbf{b}, a) = E_{\pi}(R_t \mid b_t = \mathbf{b}, a_t = a)$$

Can we find optimal Q -function with fewer data points so that we can learn from real users?

Non-parametric model for Q -function



- ▶ Belief states (from belief tracker)
- ▶ Reward – a measure of dialogue quality



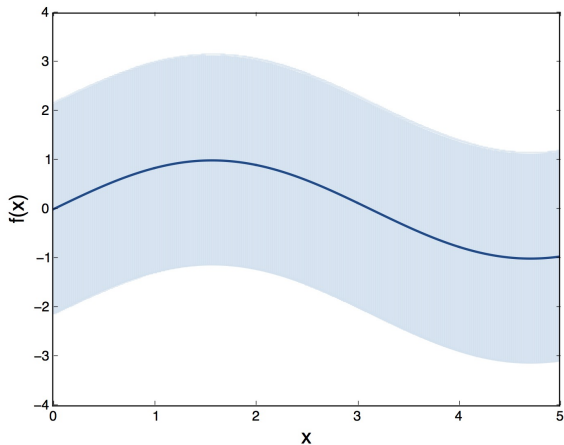
- ▶ Gaussian process model of the Q -function



- ▶ Optimal Q -function

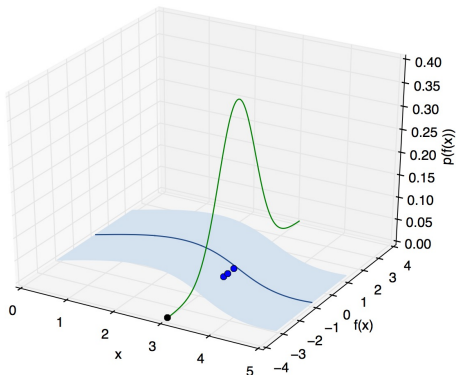
Theory: Gaussian processes prior

$$f(x) \sim \mathcal{GP}(m(x), k(x, x))$$



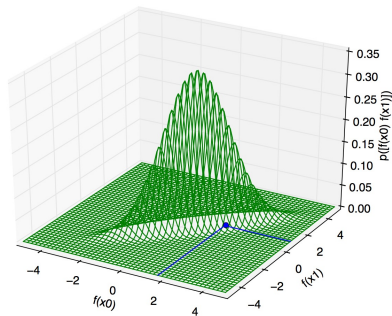
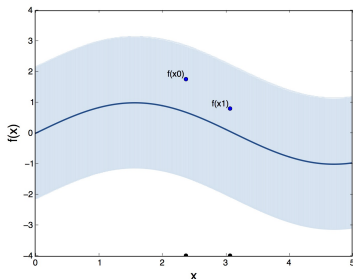
Theory: Gaussian processes kernel

$$f(x_0) \sim \mathcal{N}(m(x_0), k(x_0, x_0))$$



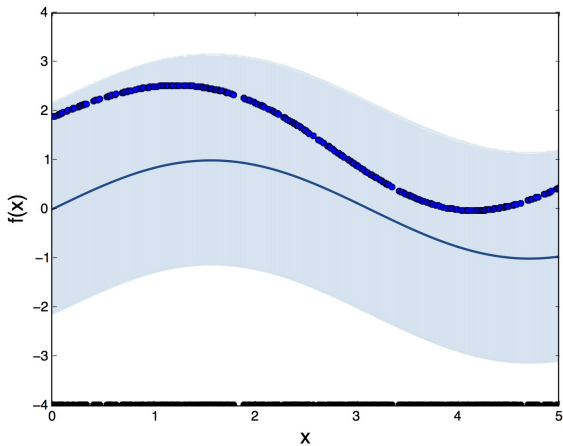
Theory: Gaussian processes kernel

$$\begin{bmatrix} f(x_0) \\ f(x_1) \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} m(x_0) \\ m(x_1) \end{bmatrix}, \begin{bmatrix} k(x_0, x_0), k(x_0, x_1) \\ k(x_1, x_0), k(x_1, x_1) \end{bmatrix} \right)$$



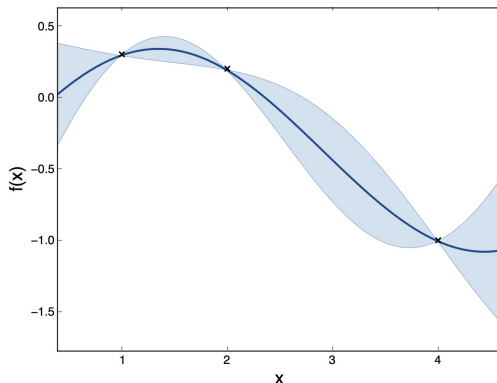
Theory: Gaussian processes kernel

Any number of function values is Gaussian distributed.



Theory: Gaussian processes posterior

- ▶ Observations \mathbf{y} in \mathbf{x} and $f(x)$ are jointly Gaussian distributed
- ▶ Conditional is then also a Gaussian process
 $f(x)|\mathbf{x}, \mathbf{y} \sim \mathcal{GP}(\bar{f}(x), \text{cov}(x, x))$

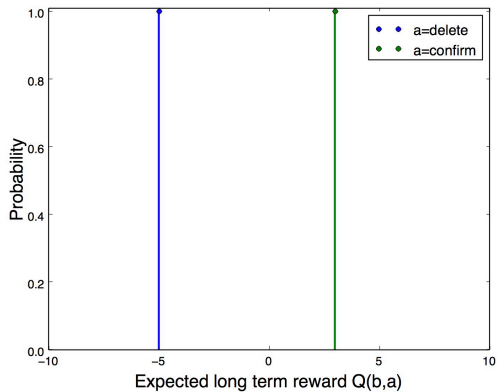
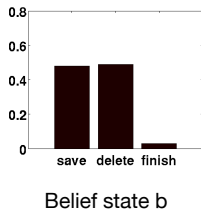


Toy dialogue problem

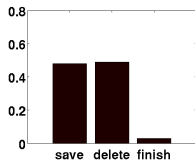
Voicemail

- ▶ States: The user wants the message saved, deleted or the dialogue is finished
- ▶ System actions: save the message, delete the message or confirm what the user wants

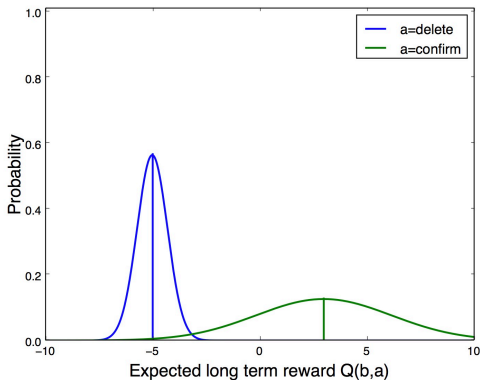
Q-function estimate without uncertainty



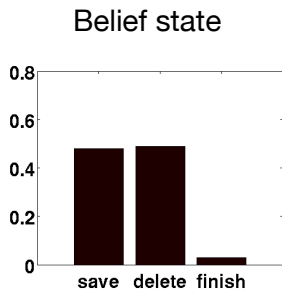
Q-function estimate with uncertainty



Belief state b



Role of the kernel function

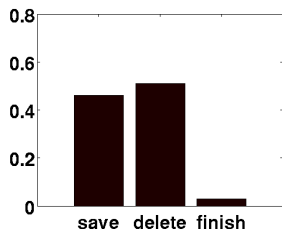


Action

Q-value

Confirm

3



Confirm



Gaussian process model for Q-function [Engel et al., 2005]

- ▶ Expected return can be expressed iteratively

$$R_t = \sum_{k=t+1}^T \gamma^{k-t-1} r_k = r_{t+1} + \gamma R_{t+1}$$

- ▶ Q-function is the expectation of the return

$$Q_{\pi}(\mathbf{b}, a) = E_{\pi} (R_t \mid b(s_t) = \mathbf{b}, a_t = a)$$

- ▶ Return can be modelled as the Q-value and residual ΔQ_{π}

$$R_t(\mathbf{b}, a) = Q_{\pi}(\mathbf{b}, a) + \Delta Q_{\pi}(\mathbf{b}, a).$$

- ▶ Relationship between immediate reward and Q-value is then:

$$r_{t+1}(\mathbf{b}, a) = Q_{\pi}(\mathbf{b}, a) - \gamma Q_{\pi}(\mathbf{b}', a') + \Delta Q_{\pi}(\mathbf{b}, a) - \gamma \Delta Q_{\pi}(\mathbf{b}', a')$$

Relationship between immediate rewards and Q-values

$$\begin{aligned}r^1 &= Q_\pi(\mathbf{b}^0, a^0) - \gamma Q_\pi(\mathbf{b}^1, a^1) \\&\quad + \Delta Q_\pi(\mathbf{b}^0, a^0) - \gamma \Delta Q_\pi(\mathbf{b}^1, a^1) \\r^2 &= Q_\pi(\mathbf{b}^1, a^1) - \gamma Q_\pi(\mathbf{b}^2, a^2) \\&\quad + \Delta Q_\pi(\mathbf{b}^1, a^1) - \gamma \Delta Q_\pi(\mathbf{b}^2, a^2) \\&\vdots \\r^t &= Q_\pi(\mathbf{b}^{t-1}, a^{t-1}) - \gamma Q_\pi(\mathbf{b}^t, a^t) \\&\quad + \Delta Q_\pi(\mathbf{b}^{t-1}, a^{t-1}) - \gamma \Delta Q_\pi(\mathbf{b}^t, a^t),\end{aligned}$$

Relationship between immediate rewards and Q-values

$$\mathbf{r}_t = \mathbf{H}_t \mathbf{q}_t^\pi + \mathbf{H}_t \Delta \mathbf{q}_t^\pi,$$

where

$$\mathbf{r}_t = [r^1, \dots, r^t]^\top$$

$$\mathbf{q}_t^\pi = [Q_\pi(\mathbf{b}^0, a^0), \dots, Q_\pi(\mathbf{b}^t, a^t)]^\top,$$

$$\Delta \mathbf{q}_t^\pi = [\Delta Q_\pi(\mathbf{b}^0, a^0), \dots, \Delta Q_\pi(\mathbf{b}^t, a^t)]^\top,$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix}.$$

Gaussian process model for Q-function

Prior $Q_\pi(\mathbf{b}, a) \sim \mathcal{GP}(0, k((\mathbf{b}, a), (\mathbf{b}, a))),$
 $\Delta Q_\pi(\mathbf{b}, a) \sim \mathcal{N}(0, \sigma^2)$

Observations Belief-action pairs $\mathbf{B}_t = [(\mathbf{b}^0, a^0), \dots, (\mathbf{b}^t, a^t)]^\top$
immediate rewards $\mathbf{r}_t = [r^1, \dots, r^t]$

Posterior $Q_\pi(\mathbf{b}, a) | \mathbf{r}_t, \mathbf{B}_t$

Posterior of the Q -function

$$\begin{aligned} Q_\pi(\mathbf{b}, a) | \mathbf{r}_t, \mathbf{B}_t &\sim \mathcal{GP}(\overline{Q}(\mathbf{b}, a), \text{cov}((\mathbf{b}, a), (\mathbf{b}, a))), \\ \overline{Q}(\mathbf{b}, a) &= \mathbf{k}_t(\mathbf{b}, a)^\top \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \sigma^2 \mathbf{H}_t \mathbf{H}_t^\top)^{-1} \mathbf{r}_t, \\ \text{cov}((\mathbf{b}, a), (\mathbf{b}, a)) &= k((\mathbf{b}, a), (\mathbf{b}, a)) \\ &\quad - \mathbf{k}_t(\mathbf{b}, a)^\top \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \sigma^2 \mathbf{H}_t \mathbf{H}_t^\top)^{-1} \mathbf{H}_t \mathbf{k}_t(\mathbf{b}, a) \end{aligned}$$

$$\begin{aligned} \mathbf{k}_t(\mathbf{b}, a) &= [k((\mathbf{b}^0, a^0), (\mathbf{b}, a)), \dots, k((\mathbf{b}^t, a^t), (\mathbf{b}, a))]^\top \\ \mathbf{K}_t &= \begin{bmatrix} k((\mathbf{b}^0, a^0), (\mathbf{b}^0, a^0)) & \cdots & k((\mathbf{b}^0, a^0), (\mathbf{b}^t, a^t)) \\ \vdots & \ddots & \vdots \\ k((\mathbf{b}^0, a^0), (\mathbf{b}^t, a^t)) & \cdots & k((\mathbf{b}^t, a^t), (\mathbf{b}^t, a^t)) \end{bmatrix} \end{aligned}$$

Applying this to an on-line setting

Computational complexity – need to invert Gram matrix \mathbf{K}_t

Sequential nature of data – need to perform updates sequentially

Kernel function – need to define correlations

GP-Sarsa algorithm

- ▶ Gram matrix is approximated with a dictionary of representative points
- ▶ Updates take place every time a reward is observed
- ▶ Kernel function is decomposed into separate kernels over belief states and actions

$$k((\mathbf{b}, a), (\mathbf{b}, a)) = k_{\mathcal{B}}(\mathbf{b}, \mathbf{b})k_{\mathcal{A}}(a, a)$$

Sparsification

- ▶ Kernel function is a dot product of potentially infinite set of feature functions $\phi(\mathbf{b}, a) = [\phi_1(\mathbf{b}, a), \phi_2(\mathbf{b}, a), \dots]^\top$

$$k((\mathbf{b}, a), (\mathbf{b}, a)) = \langle \phi(\mathbf{b}, a), \phi(\mathbf{b}, a) \rangle$$

- ▶ Gram matrix \mathbf{K}_t is approximated with Gram matrix over dictionary points $\tilde{\mathbf{K}}_t$ and coefficients $\mathbf{G}_t = [\mathbf{g}_1, \dots, \mathbf{g}_t]$

$$\mathbf{K}_t = \Phi_t^\top \Phi_t \approx \mathbf{G}_t \tilde{\mathbf{K}}_t \mathbf{G}_t^\top$$

- ▶ Dimensionality of $\tilde{\mathbf{K}}_t$ is $m \ll t$

Policy

- ▶ For given \mathbf{b} , for each action a , there is a Gaussian distribution $\hat{Q}(\mathbf{b}, a) \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a), \text{cov}((\mathbf{b}, a), (\mathbf{b}, a)))$
- ▶ Sampling from these Gaussian distributions gives Q -values $\{\hat{Q}(\mathbf{b}, a) : a \in \mathcal{A}\}$
- ▶ The highest sampled Q -value can then be selected:

$$\pi(\mathbf{b}) = \arg \max_a \left\{ \hat{Q}(\mathbf{b}, a) : a \in \mathcal{A} \right\}$$

- ▶ This balances exploration and exploitation during learning

Kernel function

Action kernel Action space is reduced to summary space and then kernel is simple δ function: $k(a, a') = \delta_a(a')$

Belief state kernel Options:

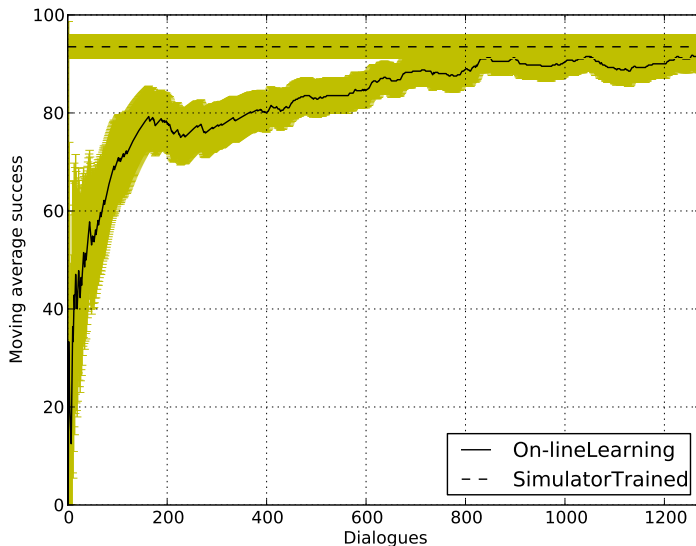
- ▶ Reduce to summary space and then calculate kernel on summary space
- ▶ Calculate the kernel directly on the full belief space
- ▶ For continuous variables use linear or Gaussian kernel

GP-Sarsa algorithm

Algorithm 1 GP-Sarsa algorithm

- 1: Define prior for Q -function
 - 2: **for** each dialogue **do**
 - 3: Initialise \mathbf{b} and choose a according to current Q estimate
 - 4: if (\mathbf{b}, a) is representative add to dictionary
 - 5: **for** each turn **do**
 - 6: Take action a observe r and next belief state \mathbf{b}'
 - 7: Choose a' according to current Q estimate
 - 8: if (\mathbf{b}', a') is representative add to dictionary
 - 9: Update posterior mean and variance of Q
 - 10: $\mathbf{b}' \rightarrow \mathbf{b}, a \rightarrow a'$
 - 11: **end for**
 - 12: **end for**
-

Learning from real users [Gasic and Young, 2014]



GPSarsa

1. Value-based or policy-based?
2. Model-based or model-free?
3. On-policy or off-policy?
4. Exploration?
5. High variance or high bias?

GPSarsa

1. Value-based
2. Model-free
3. On-policy
4. Thompson sampling
5. Biased by the choice of prior

GPSarsa - summary

- ▶ Q -function is modelled as a Gaussian process allowing posterior mean and variance to be calculated every time a reward is observed
- ▶ GP-Sarsa is a model-free, on-line algorithm which allows tractable approximation to the Gaussian process model for Q -function
- ▶ With adequate choice of the kernel function learning speed can be significantly improved
- ▶ Kernel function can be defined directly on belief space
- ▶ The bottleneck of this method is the computational complexity that is cubic in the number of representative points.

Non-parametric vs parametric approaches

- ▶ In non-parametric approaches the data are effectively the parameters of the model. The more data we have the more complex the optimisation process is.
- ▶ In parametric approaches we define the structure of the model that depends on parameters a priori and these parameters are estimated from the data.

Deep learning approaches

- ▶ Value function, Q-function or policy are approximated as neural networks
- ▶ These are approximated as non-linear functions, which is desirable in RL
- ▶ Gradient-based optimisation only finds local optima

Q-learning

For discrete space \mathcal{S} and dialogue states $s \in \mathcal{S}$

Algorithm 2 Q-learning

- 1: Initialise Q arbitrarily, $Q(\text{terminal}, \cdot) = 0$
 - 2: **repeat**
 - 3: Initialize s
 - 4: **repeat**
 - 5: Choose a ϵ -greedily
 - 6: Take action a , observe r, s'
 - 7: $Q(s, a) \leftarrow Q(s, a) + \alpha (r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
 - 8: $s \leftarrow s'$
 - 9: **until** s is terminal
 - 10: **until** convergence
-

Deep Q-network algorithm

- ▶ Q-function is approximated as a deep neural network parameterised with θ
- ▶ The gradient is given by

$$\nabla_{\theta} L(\theta) = \nabla_{\theta} (r + \gamma \max_{a'} Q(\mathbf{b}', a', \theta) - Q(\mathbf{b}, a, \theta))^2$$

DQN

1. Value-based or policy-based?
2. Model-based or model-free?
3. On-policy or off-policy?
4. Exploration?
5. High variance or high bias?

DQN

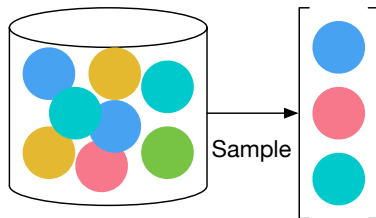
1. Value-based
2. Model-free
3. Off-policy
4. ϵ -greedy
5. Biased

Data for reinforcement learning

- ▶ In reinforcement learning data from which the agent learns is created through interaction.
- ▶ Reinforcement learning needs a lot of data, but each data point is used only once.
- ▶ In which set-up can we use the data more than once? Can we learn from experience rather than just through interaction?

Experience replay

- ▶ All interactions that the agent generates are kept in *experience replay* pool.
- ▶ The agent can sample interactions from this pool to "replay" the interactions that it had.
- ▶ This learning set-up has a foundation in neuroscience and is related to dreaming in mice.



Experience replay pool

Off-policy algorithms

- ▶ In order to apply experience replay the optimisation algorithm must be off-policy.
- ▶ Remember: off-policy learning follows a behavioural policy μ while optimising a target policy π .
- ▶ In the case of experience replay the μ is the policy that generated the experience.

Policy-based methods

- ▶ Methods that learn a parameterised policy $\pi(a|\mathbf{b}, \omega)$
- ▶ Can select actions without consulting a value function
- ▶ Optimised with respect to a performance measure $J(\omega)$

Policy gradient theorem

- ▶ $J(\omega)$ is the value of the initial belief state.

$$J(\omega) = V_{\pi}(\mathbf{b})$$

$$\nabla_{\omega} J(\omega) = E_{\pi}[\nabla_{\omega} \log \pi(a|\mathbf{b}, \omega) Q_{\pi}(\mathbf{b}, a)]$$

REINFORCE algorithm

- ▶ Policy is approximated as a deep neural network parameterised with ω
- ▶ The objective function is the value of the initial state
- ▶ The gradient is given by the policy gradient theorem where Q_π is estimated in a Monte Carlo fashion as the total return

REINFORCE

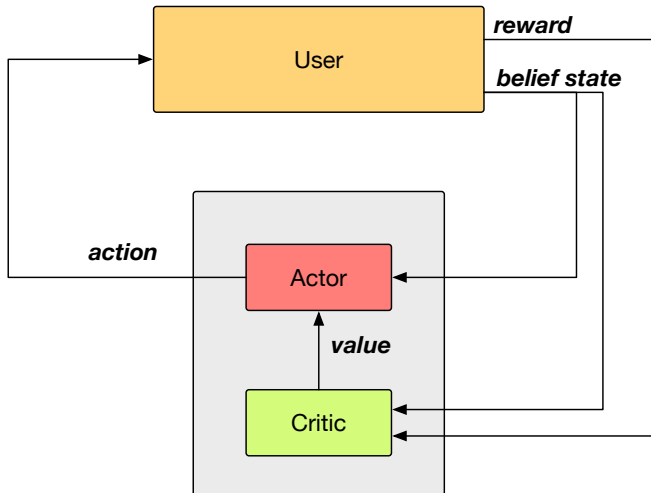
1. Value-based or policy-based?
2. Model-based or model-free?
3. On-policy or off-policy?
4. Exploration?
5. High variance or high bias?

REINFORCE

1. Policy-based
2. Model-free
3. On-policy
4. Sampling from the policy
5. High variance

Actor critic methods

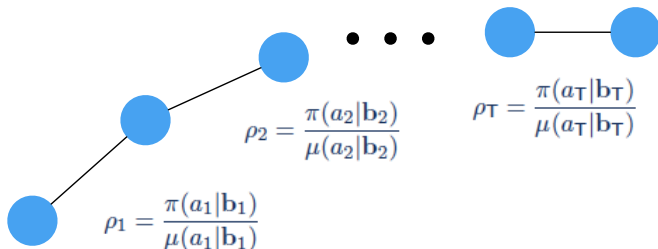
- Estimate the Q function or the value function (critic) at the same time as they estimate the policy (actor)



Importance sampling

- Importance sampling allows us to take into account that a behavioural policy produced samples while optimising the target policy.

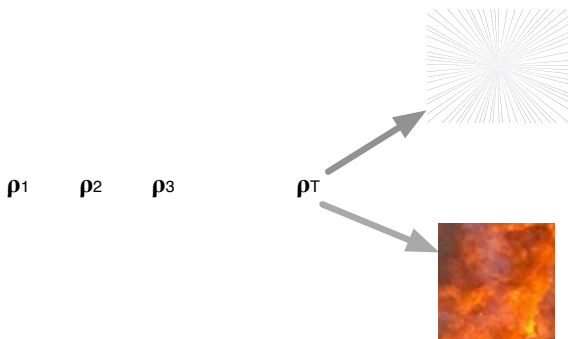
$$\rho(a|\mathbf{b}) = \frac{\pi(a|\mathbf{b})}{\mu(a|\mathbf{b})}$$



Problems with importance sampling

- ▶ Since the importance sampling ratios are unbounded some trajectories may "vanish" and some may "explode", this is why we need to truncate the importance sampling ratio.

$$E_{\pi}[R] = E_{\mu} \left[\prod_i \frac{\pi(a_i | \mathbf{b}_i)}{\mu(a_i | \mathbf{b}_i)} R \right]$$



Off-policy policy gradient theorem

- ▶ Utilise importance sampling weights to off-set that the data is generated with a behavioural policy μ .
- ▶ Use Q_π instead of R as the return in data generated by μ .
- ▶ Estimation of Q_π becomes a critic in the actor-critic framework.

Retrace - off-policy estimate for Q-function

- ▶ In order to reduce bias (of DQN for example), the estimate deploys λ -returns, a method that combines the benefits of Monte Carlo estimates (which average returns) and temporal difference learning (which base estimates on the previous estimates) by looking a few steps in the future.

Retrace - off-policy estimate for Q-function

The Retrace target is given by:

$$Q^{ret} = Q(\mathbf{b}, a, \theta) + \mathbb{E}_{\mu} \left[\sum_{t \geq 0} \gamma^t \left(\prod_{s=1}^t \lambda \min(1, \rho(a_s | \mathbf{b}_s)) \right) \left(r_t + \gamma \sum_a \pi(a | b_{t+1}) Q(\mathbf{b}_{t+1}, a, \theta) - Q(\mathbf{b}_t, a_t, \theta) \right) \right]$$

The Q-function gradient is given by

$$\nabla_{\theta} L(\theta) = \nabla_{\theta} (Q^{ret} - Q(\mathbf{b}, a, \theta))^2$$

TRPO Trust region policy optimisation

- ▶ Remember: policy is a probability distribution.
- ▶ Small changes in the parameter space can lead to erratic changes in the output policy.
- ▶ Solution: natural gradient, but expensive to compute
- ▶ Distance metric in natural gradient can be approximated as the KL divergence.
- ▶ TRPO makes sure that the KL divergence between policies of subsequent parameters is kept small.

Algorithm 3 ACER

```

1: Initialise  $\theta$  and  $\omega$  arbitrarily,  $\pi(a|\mathbf{b}, \omega)$  and  $Q_\theta(\mathbf{b}, a, \omega)$ 
2: repeat
3:   Generate episode  $e$  according to  $\pi$ 
4:   Save episode  $e$  and policy  $\pi$  in replay pool  $P$ 
5:   Sample a subset  $M$  of episodes from replay pool  $P$ 
6:   for each pair  $\mathbf{b}_{1:T}, a_{1:T}, r_{1:T}, \mu$  in  $M$  do
7:     for  $t = T$  to  $1$  do
8:        $\rho_t \leftarrow \frac{\pi(a_t|\mathbf{b}_t, \omega)}{\mu(a_t|\mathbf{b}_t)}$ 
9:        $d\omega \leftarrow d\omega + \nabla_\omega J(\omega)$ 
10:       $d\theta \leftarrow d\theta - \nabla_\theta L(\theta)$ 
11:    end for
12:  end for
13:   $k \leftarrow \nabla_\omega \text{KL}[\pi(\cdot|\omega_{pr})||\pi(\cdot|\omega)], d\omega \leftarrow d\omega - \max\{0, \frac{k^T d\omega - \delta}{\|k\|_2^2} k\}$ 
14:   $\omega \leftarrow \omega + \alpha \cdot d\omega, \theta \leftarrow \theta + \alpha \cdot d\theta$ 
15: until convergence

```

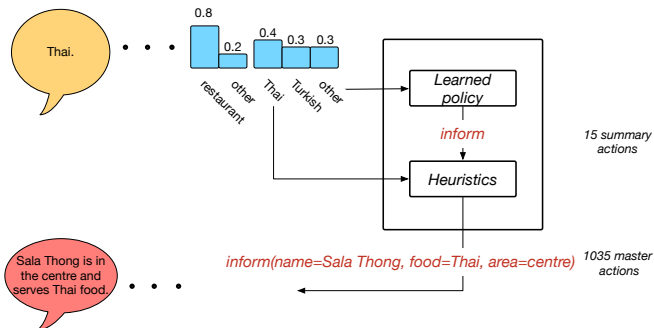
ACER

1. Value-based or policy-based?
2. Model-based or model-free?
3. On-policy or off-policy?
4. Exploration?
5. High variance or high bias?

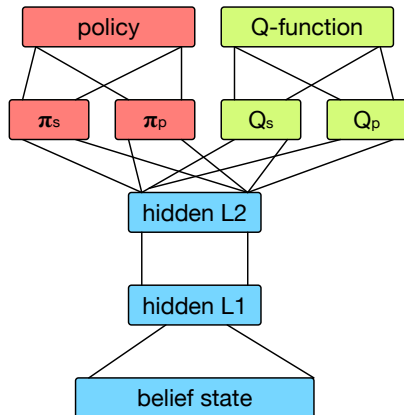
ACER

1. Actor-critic
2. Model-free
3. Off-policy
4. Thompson sampling from Boltzmann policy
5. Reduced variance and low bias

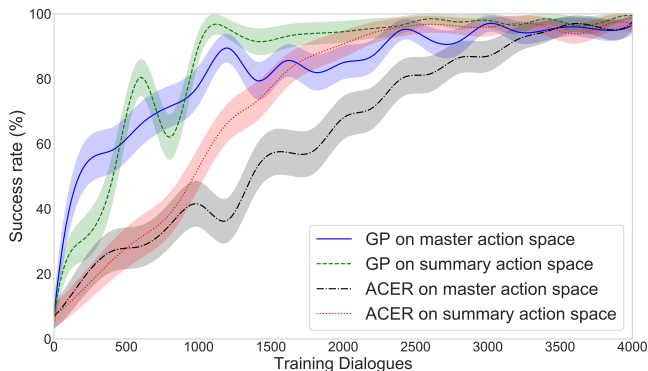
Dialogue policy: Master action space



ACER for dialogue management [Weisz et al., 2018]



ACER vs GPSARSA on summary and master space [Weisz et al., 2018]





- Note ACER needed $\sim 7h$ to train while GPSARSA needed ~ 9 days on master action space.

ACER: Summary

- ▶ ACER is an elaborate deep reinforcement learning algorithm that aims to be sample efficient by utilising experience replay.
- ▶ It utilises several methods to provide estimates with low bias and variance to support efficient learning.

References I

-  Engel, Y., Mannor, S., and Meir, R. (2005).
Reinforcement learning with Gaussian processes.
In Proceedings of ICML.
-  Gasic, M. and Young, S. (2014).
Gaussian processes for pomdp-based dialogue manager optimization.
Audio, Speech, and Language Processing, IEEE/ACM Transactions on, 22(1):28–40.
-  Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2016).
Sample efficient actor-critic with experience replay.
CoRR, abs/1611.01224.

References II



Weisz, G., Budzianowski, P., Su, P.-H., and Gasic, M. (2018).
Sample efficient deep reinforcement learning for dialogue
systems with large action spaces.
IEEE/ACM Trans. Audio, Speech and Lang. Proc.,
26(11):20832097.